Office Action dated June 5, 2002, page 2.

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983).

In this particular case, each and every feature of the presently claimed invention is not shown in *Monday* in the same arrangement as recited in the claims. For example, claim 1 reads as follows:

> 1. A method for selecting classes using a browser for use by a virtual machine in a data processing system, the method comprising:
> providing through the browser, an interface in which the interface allows for selection of classes for use by the virtual machine;
> receiving a selection of classes through the interface; and
> storing the selection of classes, wherein the selection of classes is used by the browser when initializing the virtual machine.

More specifically, *Monday* does not teach providing an interface through a browser for a selection of classes in the Java virtual machine. Additionally, the feature of storing the selection of classes such that the selection is used by the browser when initializing a virtual machine also is not shown in *Monday*.

The examiner points to the following portion of *Monday* as teaching some of the steps of the presently claimed invention:

> Referring now to FIG. **2**, sequential operations of the distributed application manager **132** start at a block **200**. As indicated at a block **202**, distributed application manager **132** displays a graphical user interface (GUI) selection screen to the user of available distributed applications to be selected by the user. For example as shown at block **202**, the user selects a Corel Word Processor, San Francisco Payroll Entry. The distributed application manager **132** checks the environment variable CLASSPATH for a set of directories to browse for the selected class file, Corel Word Processor, San Francisco Payroll Entry. If the selected x.class is located, then read the file in, parse it, and return a pointer to the requesting application. If x.class is NOT located, then a subclass of the

> CLASSLOADER, a REMOTECLASSLOADER checks if a REMOTECLASSPATH is set as indicated at a block **204**. This environment variable REMOTECLASSPATH contains a set of servers **122**, such as shown at block **202**, HTTP://WWW.IBM.COM/JAVA APPLICATIONS:HTTP//WWW.IS1.NET, ftp locations and/or machine names and directory paths on those machines, which may contain the relevant classes. Then distributed application manager **132** checks each server **122** in sequence for the particular selected x.class file as indicated at a block **204**. If the class is found, write it to the first CLASSPATH directory, thus building the class locally that the network does not have to be consulted on the next run.

Monday, column 3, lines 30-56. This portion of *Monday* does not teach providing an interface through a browser for the selection of classes. Instead, a distributed application manager is used to provide an interface. Further, the interface in *Monday* is used for loading applications and files, such as Corel Wordprocessor, San Francisco Payroll Entry, and not classes for initializing a virtual machine.

Next, the examiner points to block 304 in Figure 3 of *Monday* for the initializing step. This figure is shown as follows:

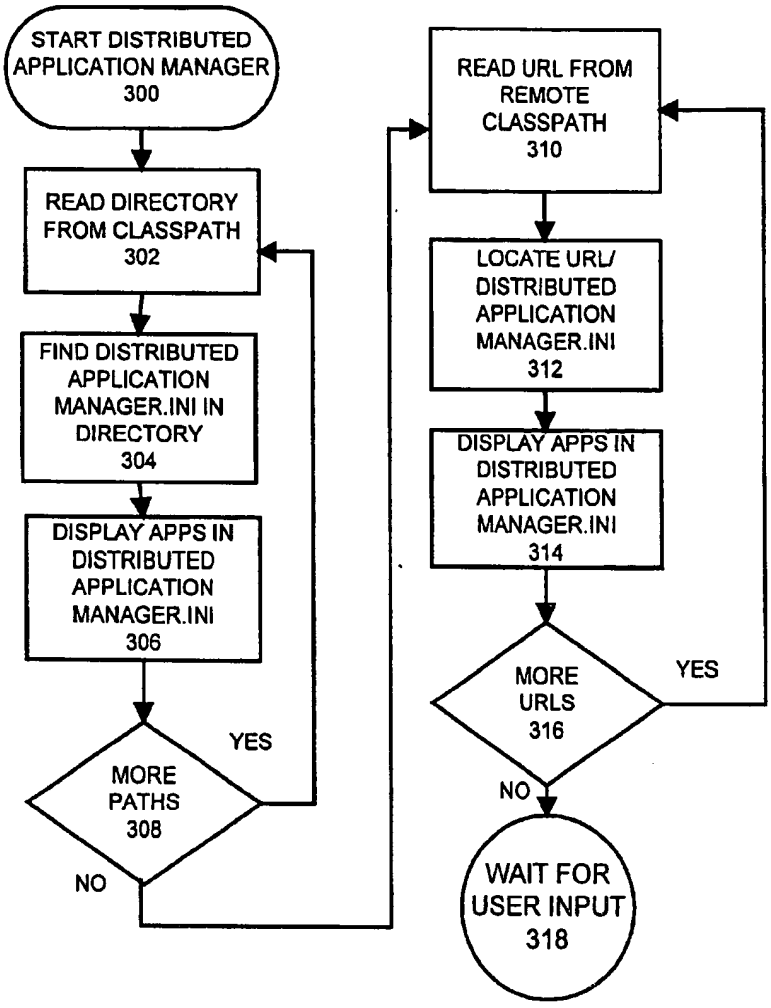

FIG.3

As can be seen, block 304 reads on a distributed application manager.ini in a directory. Nothing in this block or in the flowchart teaches initializing a virtual machine using the selection of classes stored by the method. Such a feature simply is not shown in this block or in the figure itself. This step is not taught anywhere in *Monday*.

Further, this portion of *Monday* does not teach storing the selection of classes. Instead, *Monday* writes or stores a file in a directory, rather than teaching the selection of the class from the interface. This portion of *Monday* cited by the examiner teaches determining whether the class selected by the user is present in a variable, called CLASSPATH, for a set of directories. If the class is located, the class is loaded.

Otherwise, a check is made at a number of different servers to see whether the relevant class is present. In contrast, the present invention receives a selection of classes made through an interface; the selection of the classes is stored rather than the classes themselves. This selection is then used to initialize the virtual machine.

The following portion of *Monday* includes a section cited by the examiner for teaching a virtual machine:

> Problems solved by distributed applications manager **132** are that the need to distribute and/or write an installation program for each separate distributed application is eliminated; only the parts necessary to run a particular application are downloaded, minimizing storage requirement within DASD **116**; updates to products are automatically downloaded and installed, all applications advantageously conform to JAVA or similar standards so they can be used on any platform, for example, INTERNATIONAL BUSINESS MACHINES OPERATING SYSTEM/2 ("IBM OS/2"), IBM AS/400, and MICROSOFT WINDOWS. "IBM", "OPERATING SYSTEM/2", "IBM OS/2", IBM AS/400 are trademarks of International Business Machines Corporation.

Monday, column 2, lines 55-67. This portion of *Monday* teaches that products or applications conform to Java or similar standards for use on a platform. Nowhere does this portion of *Monday* teach that the classes are used for initializing a virtual machine. This portion of *Monday* merely states that applications would conform to Java or some similar standard to allow for the applications to be used on different platforms. Nowhere does *Monday* teach using the selection of classes to initialize a virtual machine.

Next, claim 8 reads as follows:

> 8. A method for selecting classes for use by a Java virtual machine associated with a browser, the method comprising:
>
> displaying a graphical user interface in which a classpath may be selected to define classes for use with the Java virtual machine;
>
> receiving a selection of the classpath from the graphical user interface;
>
> storing the selection of the classpath; and
>
> initializing the Java virtual machine using the selection of the class path.

Claim 8 also contains features similar to those as in claim 1. Further, this claim also specifically states that a selection of a classpath is received from a graphical user interface in which this classpath is selected to define classes for use with the virtual machine. According to claim 8, the selection of the classpath is stored and the virtual machine is initialized using this selection.

In contrast, the portions of *Monday* cited by the examiner teach looking for a class file in response to a user selecting an application. The user in *Monday* does not select a classpath. Instead, the user selects a class or file. The class is associated with the class file selected by the user. In *Monday*, an environmental variable CLASSPATH is checked for a set of directories, which are to be browsed or searched for the class file. If this class file is found within the set of directories, a pointer is returned to the application. If the class is not found within the set of directories, then a check is made as to whether the class can be found in a remote location. If this class is found, then the file is written to a selected directory building the class locally. Such a feature is vastly different from selecting a classpath and storing the selection of the classpath for use in initializing a virtual machine as in claim 8.

Thus, claims 1 and 8 are not anticipated by *Monday*. The other claims are independent claims containing similar features to independent claims 1 and 8 or are dependent claims depending from one of the other claims. Thus, the other claims are patentable for the same reasons. In addition, the other claims contain other features not taught by *Monday*.

For example, claim 5 reads as follows:

> 5. The method of claim 1, wherein the step of storing the selection of classes comprises storing the selection of classes in a user profile.

The examiner points to the following section in rejecting claim 5:

> If the class is found, write it to the first CLASSPATH directory, thus building the class locally that the network does not have to be consulted on the next run.

Monday, col. 3, lines 54-56. As can be seen, *Monday* teaches storing a class in a classpath directory, while the presently claimed invention stores the selection of the classes, not the classes themselves, in a user profile. A user profile is not the same as a

directory. Instead, a user profile is a data structure. The specification describes a user profile as follows:

> With reference now to **Figure 6**, a diagram of a user profile data structure managed by a user profile manager is depicted in accordance with a preferred embodiment of the present invention. User profile data structure **600** contains information used to configure behavior of the web browser for a particular user. In the depicted example, user profile data structure **600** includes a profile name **602**, a Java class path **604**, Java parameters **606**, a Java path **608**, and a Java class path option **610**. Profile name **602** is used to uniquely identify the profile from other profiles when the browser contains multiple user profiles. Java class path **604** is used to identify the path in which classes are loaded for use by the JVM. Java parameters **606** contain parameters used by a JVM when the browser initializes or starts a JVM for use with the browser. These parameters may include, for example, initial heap size, garbage collection information, Java stack size, and reporting options for JVM information. Java path **608** includes the path and file name for the JVM that is to be used with the browser. Java class path option **610** provides information that may be used to depend an extended class path to the beginning or end of the system defined class path.

Specification, page 16, line 25-page 17, line 17. Thus, a directory is not the same as a user profile. Therefore, claims 1-19 are not anticipated by *Monday*.

Further, these claims are not obvious in view of *Monday*. No teaching, suggestion, or incentive is present for making the necessary modifications to *Monday* to reach the presently claimed invention. One of ordinary skill in the art would not make the modifications when *Monday* is considered as a whole. For example, when considering *Monday* as a whole, one of ordinary skill in the art would consider the problem addressed or recognized. *Monday* recognizes the following:

> The ability to install and run an application without an installation utility customized for the application was virtually unheard of until recently. Java Applets run within an appletviewer or a web browser have set a precedence for the invisible insulation of a client applet (tiny application). The browser asks the server to download an applet by name, once the executable code or the class file is downloaded, the applet is free to run within the confines of the Web Browser.
>
> Both Java Applets and Java Applications can run on various client machines without modification to any computing platform, thus saving the costs associated with developing software for multiple platforms. The Java programming language is a simple, object-oriented, network-savvy, interpreted, robust, secure, architecture neutral, portable, high-

> performance, multithreaded, dynamic language. There are many distinctions between Java Applets and Java Applications. An appletviewer or web browser do not support Java Applications.
>
> A need exists for an effective technique for managing distributed applications. It is desirable to eliminate the need to distribute and/or write an installation program for each separate distributed application.

Monday, column 1, lines 13-33. *Monday* is directed towards a problem associated with the current practice of distributing or writing a installation program for each separate distributed application.

In contrast, the presently claimed invention is directed towards the following problem:

> Presently available browsers are designed with a notion of a fixed JVM, which uses a fixed value for the Java classpath. No flexible mechanisms are presently available for modifying or viewing the classpath within a browser. The problem with this approach is that of the global nature of the changing system classpath requires the user to modify the system defined global classpath variable manually or via a script which is executed prior to executing the browser. This system classpath variable is shared with other non-Java browser related applets or applications in addition to the Java enabled browser. Another problem exists for browsers that provide for a multi-user environment in which multiple users/user profiles are being employed. In such a situation, each user profile is forced to use the same environment.
>
> Therefore, it would be advantageous to have an improved method and apparatus for providing users an ability to use more recent versions of JVMs without having to wait for an updated version of the web browser.

Specification, page 5, lines 3-21. The present invention recognizes problems associated with flexibility in using different versions of a Java virtual machine. The present invention solves this problem through allowing the selection of classes through a user interface in a browser and the storing of the selection for use in initializing a virtual machine, such as a Java virtual machine.

Monday provides a different solution when viewed as a whole by one of ordinary skill in the art. *Monday* specifically teaches:

> In brief, a method and computer program product for managing distributed applications on a local computer system, and a distributed application manager are provided. The distributed application manager running at a local computer system keeps a list of available distributed

> applications as well as a list of servers from which these distributed applications can be downloaded. The distributed application manager presents a selection screen to the user, containing a list of available distributed applications; and based on a user selection from the list, the distributed application manager searches for the selected application on a path of servers and a path of directories in each server. When the distributed application manager finds the selected application, it downloads the application from the server, installs the selected application at the local computer system and invokes the application for the user.

Monday, column 1, lines 48-63. In viewing *Monday* as a whole, *Monday* is directed towards a system to allow a user to select distributed applications with a distributed application manager searching for the applications on a path of servers and a path of directories on each server. When found, the selected application is downloaded, installed, and then invoked for the user. Thus, *Monday* is directed towards an entirely different problem and solution than that of the presently claimed invention. Therefore, one of ordinary skill in the art would not interpret or modify *Monday* in the manner necessary to reach the presently claimed invention.

Therefore, the rejection of claims 1-19 under 35 U.S.C. § 102 has been overcome.

II. 35 U.S.C. § 103, Obviousness

The examiner has rejected claims 10 and 11 under 35 U.S.C. § 103 as being unpatentable over *Monday*. This rejection is respectfully traversed.

Claims 10 and 11 are dependent claims depending from claim 8. Thus, these claims are patentable for the same reasons as stated above for independent claim 8. Even if *Monday* taught the appending step in claims 10 and 11, such a modification of *Monday* would not reach the presently claimed invention for the reasons stated above. Therefore, the rejection of claims 10 and 11 under 35 U.S.C. § 103 has been overcome.

III. Conclusion

It is respectfully urged that the subject application is patentable over the cited reference and is now in condition for allowance.

The examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: 8/29/02

Respectfully submitted,

Duke W. Yee
Reg. No. 34,285
Carstens, Yee & Cahoon, LLP
P.O. Box 802334
Dallas, TX 75380
(972) 367-2001
Attorney for Applicants